



## 编辑推荐

针对 Java SE6 平台进行了全面更新，涵盖 Java 语言核心内容。大量精心设计代码示例。CSDN Java 大版主隆重推荐。

对于想将 Java 应用于实际项目中的程序员来说，本书是一本权威性的指导书籍。本书针对 JavaSE6 平台进行了全面更新，并通过大量测试过的示例说明了最重要的语言特性和类库特性。本书示例程序经过精心地设计，不但具有实用价值，而

且易阅读、易理解，可以作为读者自己编写程序的良好开端。

本书试图让读者快速地了解 JavaSE6 的新特性，并帮助读者有效地从 Java 早期版本升级到最新版本，或从一种其他的语言变换为使用 Java 语言。作者十分注重 Java 语言的基本概念以及用户界面程序设计基础。本卷详细介绍以下内容：

- Java 语言基础知识
- 面向对象程序设计
- 接口与内部类
- 事件监听器模型
- Swing 图形用户界面程序设计
- 打包应用程序
- 异常处理
- 登录与调试
- 泛型程序设计
- 集合框架
- 多线程

有关 XML 处理、网络、数据库、本地方法、安全、高级 AWT/Swing 和其他高级特性请参阅《Java 核心技术，卷 II：高级特性（原书第 8 版）》。

## 内容简介

《Java 核心技术》出版以来一直畅销不衰，深受读者青睐，每个新版本都尽可能地跟上 Java 开发工具箱发展的步伐，而且每一版都重新改写了部分内容，以便适应 Java 的最新特性。本版也不例外，它反映了 Java SE 6 的新特性。全书共 14 章，包括 Java 基本的程序结构、对象与类、继承、接口与内部类、图形程序设计、事件处理、Swing 用户界面组件、部署应用程序和 Applet、异常日志断言和调试、泛型程序设计、集合以及多线程等内容。

全书对 Java 技术的阐述精确到位，叙述方式深入浅出，并包含大量示例，从而帮助读者充分理解 Java 语言以及 Java 类库的相关特性。

## 作者简介

Cay S. Horstmann 参与编写《Core Java Server Faces》第 2 版（Prentice Hall, 2007）。Cay 是圣何塞州立大学计算机科学系教授、Java 的倡导者，并经常在计算机会议上讲演。

Cray Cornell 已经编写并教授程序设计专业课程 20 余年，是 Apress 的创始人。他撰写的程序设计专业书籍十分畅销，是 Jolt Award 的最终获奖者之一，并通过 Visual Basic 资料赢得了 Readers Choice。

## 目录

## 第1章 Java 程序设计概述

### 1.1 Java 程序设计平台

### 1.2 Java “白皮书”的关键术语

#### 1.2.1 简单性

#### 1.2.2 面向对象

#### 1.2.3 网络技能

#### 1.2.5 安全性

#### 1.2.6 体系结构中立

#### 1.2.7 可移植性

#### 1.2.8 解释型

#### 1.2.9 高性能

#### 1.2.10 多线程

#### 1.2.11 动态性

### 1.3 Java Appletc 与 Internet

### 1.4 Java 发展简史

### 1.5 关于 Java 的常见误解

## 第2章 Java 程序设计环境

### 2.1 安装 Java 开发工具箱

#### 2.1.1 下载 JDK

#### 2.1.2 设置执行路径

#### 2.1.3 安装源代码库和文档

#### 2.1.4 安装本书中的示例

#### 2.1.5 导航 Java 目录

### 2.2 选择开发环境

### 2.3 使用命令行工具

### 2.4 使用集成开发环境

### 2.5 运行图形化应用程序

### 2.6 建立并运行 appletc

## 第3章 Java 基本的程序设计结构

### 3.1 一个简单的 Java 应用程序

### 3.2 注释

### 3.3 数据类型

#### 3.3.1 整型

#### 3.3.2 浮点类型

#### 3.3.3 char 类型

#### 3.3.4 boolean 类型

### 3.4 变量

#### 3.4.1 变量初始化

#### 3.4.2 常量

### 3.5 运算符

#### 3.5.1 自增运算符与自减运算符

#### 3.5.2 关系运算符与 boolean 运算符

#### 3.5.4 数学函数与常量

#### 3.5.5 数值类型之间的转换

- 3.5.6 强制类型转换
  - 3.5.7 括号与运算符级别
  - 3.5.8 枚举类型
- 3.6 字符串
  - 3.6.1 子串
  - 3.6.2 拼接
  - 3.6.3 不可变字符串
  - 3.6.4 检测字符串是否相等
  - 3.6.5 代码点与代码单元
  - 3.6.6 字符串 API
  - 3.6.7 阅读联机 API 文档
  - 3.6.8 构建字符串
- 3.7 输入输出
  - 3.7.1 读取输入
  - 3.7.2 格式化输出
  - 3.7.3 文件输入与输出
- 3.8 控制流程
  - 3.8.1 块作用域
  - 3.8.2 条件语句
  - 3.8.3 循环
  - 3.8.4 确定循环
  - 3.8.5 多重选择: switch 语句
  - 3.8.6 中断控制流程语句
- 3.9 大数值
- 3.10 数组
  - 3.10.1 for 循环
  - 3.10.2 数组初始化以及匿名数组
  - 3.10.4 命令行参数
  - 3.10.5 数组排序
  - 3.10.6 多维数组
  - 3.10.7 不规则数组
- 第 4 章 对象与类
  - 4.1 面向对象程序设计概述
    - 4.1.1 类
    - 4.1.2 对象
    - 4.1.3 识别类
    - 4.1.4 类之间的关系
  - 4.2 使用现有类
    - 4.2.1 对象与对象变量
    - 4.2.2 Java 类库中的 GregorianCalendar 类
    - 4.2.3 更改器方法与访问器方法
  - 4.3 用户自定义类
    - 4.3.1 一个 Employee 类
    - 4.3.2 多个源文件的使用

- 4.3.3 解析 Employee 类
    - 4.3.4 从构造器开始
    - 4.3.5 隐式参数与显式参数
    - 4.3.6 封装的优点
    - 4.3.7 基于类的访问权限
    - 4.3.8 私有方法
    - 4.3.9 Final 实例域
  - 4.4 静态域与静态方法
    - 4.4.1 静态域
    - 4.4.2 静态常量
    - 4.4.3 静态方法
    - 4.4.4 Factory 方法
    - 4.4.5 Main 方法
  - 4.5 方法参数
  - 4.6 对象构造
    - 4.6.1 重载
    - 4.6.2 默认域初始化
    - 4.6.3 默认构造器
    - 4.6.4 显式域初始化
    - 4.6.5 参数名
    - 4.6.6 调用另一个构造器
    - 4.6.7 初始化块
    - 4.6.8 对象析构与 finalize 方法
  - 4.7 包
    - 4.7.1 类的导入
    - 4.7.2 静态导入
    - 4.7.3 将类放入包中
    - 4.7.4 包作用域
  - 4.8 类路径
  - 4.9 文档注释
    - 4.9.1 注释的插入
    - 4.9.2 类注释
    - 4.9.3 方法注释
    - 4.9.4 域注释
    - 4.9.5 通用注释
    - 4.9.6 包与概述注释
    - 4.9.7 注释的抽取
  - 4.10 类设计技巧
- 第 5 章 继承
- 5.1 类.c 超类和子类
    - 5.1.1 继承层次
    - 5.1.2 多态
    - 5.1.3 动态绑定
    - 5.1.4 阻止继承: final 类和方法

- 5.1.5 强制类型转换
- 5.1.6 抽象类
- 5.1.7 受保护访问
- 5.2 Object: 所有类的超类
  - 5.2.1 Equals 方法
  - 5.2.2 相等测试与继承
  - 5.2.3 hashCode 方法
  - 5.2.4 ToString 方法
- 5.3 泛型数组列表
  - 5.3.1 访问数组列表元素
  - 5.3.2 类型化与原始数组列表的兼容性
- 5.4 对象包装器与自动打包
- 5.5 参数数量可变的方法
- 5.6 枚举类
- 5.7 反射
  - 5.7.1 Class 类
  - 5.7.2 捕获异常
  - 5.7.3 利用反射分析类的能力
  - 5.7.4 在运行时使用反射分析对象
  - 5.7.5 使用反射编写泛型数组代码
  - 5.7.6 方法指针
- 5.8 继承设计的技巧
- 第 6 章 接口与内部类
  - 6.1 接口
    - 6.1.1 接口的特性
    - 6.1.2 接口与抽象类
  - 6.2 对象克隆
  - 6.3 接口与回调
  - 6.4 内部类
    - 6.4.1 使用内部类访问对象状态
    - 6.4.2 内部类的特殊语法规则
    - 6.4.3 内部类是否有用.c 必要和安全
    - 6.4.4 局部内部类
    - 6.4.5 由外部方法访问 final 变量
    - 6.4.6 匿名内部类
    - 6.4.7 静态内部类
  - 6.5 代理
- 第 7 章 图形程序设计
  - 7.1 Swing 概述
  - 7.2 创建框架
  - 7.3 框架定位
  - 7.4 框架属性
  - 7.5 决定框架大小
  - 7.6 在组件中显示信息

- 7.7 2D 图形
- 7.8 颜色
- 7.9 为文本设定特殊字体
- 7.10 图像
- 第8章 事件处理
  - 8.1 事件处理基础
    - 8.1.1 实例：处理按钮点击事件
    - 8.1.2 建议使用内部类
    - 8.1.3 创建包含一个方法调用的监听器
    - 8.1.4 实例：改变观感
    - 8.1.5 适配器类
  - 8.2 动作
  - 8.3 鼠标事件
  - 8.4 AWT 事件继承层次
- 第9章 Swing 用户界面组件
  - 9.1 Swing 和模型-视图-控制器设计模式
    - 9.1.1 设计模式
    - 9.1.2 模型-视图-控制器模式
    - 9.1.3 Swing 按钮的模型-视图-控制器分析
  - 9.2 布局管理器概述
    - 9.2.1 边框布局
    - 9.2.2 网格布局
  - 9.3 文本输入
    - 9.3.1 文本域
    - 9.3.2 标签和标签组件
    - 9.3.3 密码域
    - 9.3.4 文本区
    - 9.3.5 滚动窗格
  - 9.4 选择组件
    - 9.4.1 复选框
    - 9.4.2 单选按钮
    - 9.4.3 边框
    - 9.4.4 组合框
    - 9.4.5 滑块
  - 9.5 菜单
    - 9.5.1 菜单创建
    - 9.5.2 菜单项中的图标
    - 9.5.3 复选框和单选按钮菜单项
    - 9.5.4 弹出菜单
    - 9.5.5 快捷键和加速器
    - 9.5.6 启用和禁用菜单项
    - 9.5.7 工具栏
    - 9.5.8 工具提示
  - 9.6 复杂的布局管理

- 9.6.1 网格组布局
  - 9.6.2 组布局
  - 9.6.3 不使用布局管理器
  - 9.6.4 定制布局管理器
  - 9.6.5 遍历顺序
- 9.7 对话框
  - 9.7.1 选项对话框
  - 9.7.2 创建对话框
  - 9.7.3 数据交换
  - 9.7.4 文件对话框
  - 9.7.5 颜色选择器
- 第 10 章 部署应用程序和 applet
  - 10.1 JAR 文件
    - 10.1.1 清单文件
    - 10.1.2 可运行 JAR 文件
    - 10.1.3 资源
    - 10.1.4 密封
  - 10.2 Java WebStart
    - 10.2.1 沙箱
    - 10.2.2 签名代码
    - 10.2.3 JNLPcAPI
  - 10.3 Applet
    - 10.3.1 一个简单的 caplet
    - 10.3.2 将应用程序转换为 applet
    - 10.3.3 Applet 的 HTMLc 标记和属性
    - 10.3.4 Objectc 标记
    - 10.3.5 使用参数向 applet 传递信息
    - 10.3.6 访问图像和音频文件
    - 10.3.7 Applet 上下文
  - 10.4 应用程序存储的配置
    - 10.4.1 属性映射
    - 10.4.2 PreferencescAPI
- 第 11 章 异常.c 日志.c 断言和调试
  - 11.1 处理异常
    - 11.1.1 异常分类
    - 11.1.2 声明已检查异常
    - 11.1.3 如何抛出异常
    - 11.1.4 创建异常类
  - 11.2 捕获异常
    - 11.2.1 捕获多个异常
    - 11.2.2 再次抛出异常与异常链
    - 11.2.3 Finally 子句
    - 11.2.4 分析堆栈跟踪元素
  - 11.3 使用异常机制的建议



- 11.4 断言
  - 11.4.1 启用和禁用断言
  - 11.4.2 使用断言的建议
  - 11.4.3 为文档使用断言
- 11.5 记录日志
  - 11.5.1 基本日志
  - 11.5.2 高级日志
  - 11.5.3 修改日志管理器配置
  - 11.5.4 本地化
  - 11.5.5 处理器
  - 11.5.6 过滤器
  - 11.5.7 格式化器
  - 11.5.8 日志记录说明
- 11.6 调试技术
  - 11.6.1 使用控制台窗口
  - 11.6.2 跟踪 AWT 事件
  - 11.6.3 AWT 的 Robot 类
- 11.7 使用调试器
- 第 12 章 泛型程序设计
  - 12.1 为什么要使用泛型程序设计
  - 12.2 简单泛型类的定义
  - 12.4 类型变量的限定
  - 12.5 泛型代码和虚拟机
    - 12.5.1 翻译泛型表达式
    - 12.5.2 翻译泛型方法
    - 12.5.3 调用遗留代码
  - 12.6 约束与局限性
    - 12.6.1 不能用基本类型实例化类型参数
    - 12.6.2 运行时类型查询只适用于原始类型
    - 12.6.3 不能抛出也不能捕获泛型类实例
    - 12.6.4 参数化类型的数组不合法
    - 12.6.5 不能实例化类型变量
    - 12.6.6 泛型类的静态上下文中类型变量无效
    - 12.6.7 注意擦除后的冲突
  - 12.7 泛型类型的继承规则
  - 12.8 通配符类型
    - 12.8.1 通配符的超类型限定
    - 12.8.2 无限定通配符
    - 12.8.3 通配符捕获
  - 12.9 反射和泛型
    - 12.9.1 使用 ClassCtC 参数进行类型匹配
- 第 13 章 集合
  - 13.1 集合接口
    - 13.1.1 将集合的接口与实现分离

- 13.1.2 Java 类库中的集合接口和迭代器接口
- 13.2 具体的集合
  - 13.2.1 链表
  - 13.2.2 数组列表
  - 13.2.3 散列集
  - 13.2.4 树集
  - 13.2.5 对象的比较
  - 13.2.6 队列与双端队列
  - 13.2.7 优先级队列
  - 13.2.8 映射表
  - 13.2.9 专用集与映射表类
- 13.3 集合框架
  - 13.3.1 视图与包装器
  - 13.3.2 批操作
  - 13.3.3 集合与数组之间的转换
- 13.4 算法
  - 13.4.1 排序与混排
  - 13.4.2 二分查找
  - 13.4.3 简单算法
  - 13.4.4 编写自己的算法
- 13.5 遗留的集合
  - 13.5.1 Hashtable 类
  - 13.5.2 枚举
  - 13.5.3 属性映射表
  - 13.5.4 栈
  - 13.5.5 位集
- 第 14 章 多线程
  - 14.1 线程的概念
  - 14.2 中断线程
  - 14.3 线程状态
    - 14.3.1 新生线程
    - 14.3.2 可运行线程
    - 14.3.3 被阻塞线程和等待线程
    - 14.3.4 被终止的线程
  - 14.4 线程属性
    - 14.4.1 线程优先级
    - 14.4.2 守护线
    - 14.4.3 未捕获异常处理器
  - 14.5 同步
    - 14.5.1 竞争条件的一个例子
    - 14.5.2 详解竞争条件
    - 14.5.3 锁对象
    - 14.5.4 条件对象
    - 14.5.5 synchronized 关键字

- 14.5.6 同步阻塞
- 14.5.7 监视器概念
- 14.5.8 Volatile 域
- 14.5.9 死锁
- 14.5.10 锁测试与超时
- 14.5.11 读/写锁
- 14.5.12 为什么弃用 stop 和 suspend 方法
- 14.7 线程安全的集合
  - 14.7.1 高效的映像.c 集合和队列
  - 14.7.2 写数组的拷贝
  - 14.7.3 旧的线程安全的集合
- 14.8 Callable 与 Future
- 14.9 执行器
  - 14.9.1 线程池
  - 14.9.2 预定执行
  - 14.9.3 控制任务组
- 14.10 同步器
  - 14.10.1 信号量
  - 14.10.2 倒计时门栓
  - 14.10.3 障栅
  - 14.10.4 交换器
  - 14.10.5 同步队列
  - 14.10.6 例子：暂停动画与恢复动画
- 14.11 线程与
  - 14.11.1 运行耗时的任务
  - 14.11.2 使用 Swing 工作器
  - 14.11.3 单一线程规则

## 书摘插图

### 第1章 Java 程序设计概述

- ▲Java 程序设计平台
- ▲Java “白皮书”的关键术语
- ▲Java 与 Internet
- ▲Java 发展简史
- ▲关于 Java 的常见误解

1996 年 Java 第一次发布就引起了人们的极大兴趣。关注 Java 的人士不仅限于计算机出版界，还有诸如《纽约时报》、《华盛顿邮报》、《商业周刊》这样的主流媒体。Java 是第一种也是惟一的一种在 National Public Radio 上占用了 10 分钟时间进行介绍的程序设计语言，并且还得到了 \$100000000 的风险投资基金。这些基金全部用来支持用这种特别的计算机语言开发的产品。重温那些令人兴奋的日子是很有意思的。本章将简要地介绍一下 Java 语言的发展历史。

#### 1.1 Java 程序设计平台

本书的第1版是这样描写Java的：“作为一种计算机语言，Java的广告词确实有点夸大其辞。然而，Java的确是一种优秀的程序设计语言。作为一个名副其实的程序设计人员，使用Java无疑是一个好的选择。有人认为：Java将有望成为一种最优秀的程序设计语言，但还需要一个相当长的发展时期。一旦一种语言应用于某个领域，与现存代码的相容性问题就摆在了人们的面前。”

我们的编辑手中有许多这样的广告词。这是Sun公司高层的某位不愿透露姓名的人士提供的。然而，现在看起来，当初的这些预测还是有一定准确性的。Java有许多非常优秀的语言特性，本章稍后将会详细地讨论这些特性。由于相容性这个严峻的问题确实存在于现实中，所以，或多或少地还是有一些“累赘”被加到语言中，这就导致Java并不如想像中的那么完美无瑕。

但是，正像我们在第1版中已经指出的那样，Java并不只是一种语言。在此之前出现的那么多种语言也没有能够引起那么大的轰动。Java是一个完整的平台，有一个庞大的库，其中包含了很多可重用的代码和一个提供诸如安全性、跨操作系统的可移植性以及自动垃圾收集等服务的执行环境。

作为一名程序设计人员，常常希望能够有一种语言，它具有令人赏心悦目的语法和易于理解的语义（C++不是这样的）。与许多其他的优秀语言一样，Java恰恰满足了这些要求。有些语言提供了可移植性、垃圾收集器等等，但是，没有提供一个大库。如果想要有奇特的绘图功能、网络连接功能和数据库存取功能就必须自己动手编写代码。Java这种功能齐全的出色语言，具有高质量的执行环境以及庞大的库。正是因为它集多种优势于一身，所以对广大的程序设计人员有着不可抗拒的吸引力。

.....

书摘与插图



[下载以后 点击此处可查看更多内容](#)